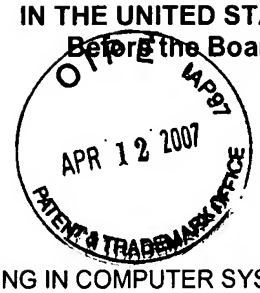
**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**
**Before the Board of Patent Appeals and Interferences**




In re Patent Application of: NEVILL
Serial No. 10/781,867
Filed: February 20, 2004
Title: MEMORY RECYCLING IN COMPUTER SYSTEMS

Atty Dkt. SCS-550-513
C# M#
TC/A.U.: 2188
Examiner: M. McFadden
Date: April 12, 2007

**Mail Stop Appeal Brief - Patents**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

☐ **Correspondence Address Indication Form Attached.**

☐ **NOTICE OF APPEAL**
Applicant hereby **appeals** to the Board of Patent Appeals and Interferences from the last decision of the Examiner twice/finally rejecting applicant's claim(s). $500.00 (1401)/$250.00 (2401) $

☒ An appeal **BRIEF** is attached in the pending appeal of the above-identified application $500.00 (1402)/$250.00 (2402) $ 500.00

☐ Credit for fees paid in prior appeal without decision on merits -$ ( )

☐ A reply brief is attached. **(no fee)**

☐ Petition is hereby made to extend the current due date so as to cover the filing date of this paper and attachment(s)
One Month Extension $120.00 (1251)/$60.00 (2251)
Two Month Extensions $450.00 (1252)/$225.00 (2252)
Three Month Extensions $1020.00 (1253/$510.00 (2253)
Four Month Extensions $1590.00 (1254/$795.00 (2254) $

☐ "Small entity" statement attached.

Less month extension previously paid on -$( )

**TOTAL FEE ENCLOSED** $ 500.00

Any future submission requiring an extension of time is hereby stated to include a petition for such time extension. The Commissioner is hereby authorized to charge any deficiency, or credit any overpayment, in the fee(s) filed, or asserted to be filed, or which should have been filed herewith (or with any paper hereafter filed in this application by this firm) to our **Account No. 14-1140.** A duplicate copy of this sheet is attached.

901 North Glebe Road, 11th Floor
Arlington, Virginia 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100
SCS:kmm

NIXON & VANDERHYE P.C.
By Atty: Stanley C. Spooner, Reg. No. 27,393

Signature: _____

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**
**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

| | |
|---|---|
| In re Patent Application of | Confirmation No.: 5181 |
| NEVILL | Atty. Ref.: 550-513 |
| Serial No. 10/781,867 | Group: 2188 |
| Filed: February 20, 2004 | Examiner: M. McFadden |

For: MEMORY RECYCLING IN COMPUTER SYSTEMS

* * * * * * * * * * * * * * * * * * * * * * * * *

# APPEAL BRIEF

On Appeal From Group Art Unit 2188

Stanley C. Spooner
**NIXON & VANDERHYE P.C.**
11th Floor, 901 North Glebe Road
Arlington, Virginia 22203
(703) 816-4028
Attorney for Appellant

# TABLE OF CONTENTS

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**
**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In re Patent Application of

NEVILL — Atty. Ref.: 550-513

Serial No. 10/781,867 — Group: 2188

Filed: February 20, 2004 — Examiner: M. McFadden

For: MEMORY RECYCLING IN COMPUTER SYSTEMS

* * * * * * * * * * * * * * * * * * * * * * * *

April 12, 2007

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

## APPEAL BRIEF

Sir:

## I. REAL PARTY IN INTEREST

The real party in interest in the above-identified appeal is ARM Limited by virtue of an assignment of rights from the inventor to ARM Limited recorded February 20, 2004 at Reel 15010, Frame 602.

## II. RELATED APPEALS AND INTERFERENCES

There are believed to be no related appeals, interferences or judicial proceedings with respect to the present application, other than the Pre-Appeal Brief Request for Review previously filed in this appeal.

## III. STATUS OF CLAIMS

Claims 1-33 stand variously rejected in the Final Official Action. The Examiner contends that all claims are either anticipated under 35 USC §102 by Wilson ("Uniprocessor Garbage Collection Techniques") or are obvious under 35 USC §103 over Wilson by itself or in combination with Hosoya ("Garbage Collection via Dynamic Type Interface"). The above rejections of claims 1-33 are appealed.

## IV. STATUS OF AMENDMENTS

No further response has been submitted with respect to the Final Official Action in this application other than the filing of a Pre-Appeal Brief Request for Review which decision was mailed March 12, 2007 (Paper No. 20070308).

## V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Appellants' specification and figures provide an explanation of the claimed invention set out in independent claims 1, 11, 21 and 31-33, with each claimed structure addressed as to its location in the specification and in the figures.

"1. A method of controlling execution of a processing task within a data processing system, said method comprising the steps of:

executing [step 120 shown in Figure 1 and discussed on page 11, lines 9-30 and elsewhere in the specification] said processing task including allocating memory areas for data storage; and

suspending [step 130 shown in Figure 1 and discussed on page 11, lines 9-30 and elsewhere in the specification] an actual execution path of said processing task at an execution point to perform memory management, said memory management comprising the steps of:

identifying [step 140 shown in Figure 1 and discussed on page 11, lines 9-30, step 330 shown in Figure 3 and discussed on page 16, lines 4-22, step 550 shown in Figure 5 and discussed on page 18, lines 4-23 and elsewhere in the specification] at least one data item roots occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

determining [step 150 shown in Figure 1 and discussed on page 11, lines 9-30, step 340 shown in Figure 3 and discussed on page 16, lines 4-22 and elsewhere in the specification] a correlation between reference values corresponding to said at least one data item roots and memory areas allocated during said execution up to said execution point by identifying at least one data item reachable from said at least one data item roots; and

performing [step 160 shown in Figure 1 and discussed on page 11, lines 9-30 and on page 13, lines 5-17, step 350 shown in Figure 3 and discussed on page

16, lines 4-22 and elsewhere in the specification] a memory management operation on allocated memory areas in dependence upon said correlation."

"11. A computer program product comprising a computer readable storage medium comprising computer readable instructions that, when executed, cause a computer to control execution of a processing task within a data processing system, said computer program comprising:

execution code operable to execute [step 120 shown in Figure 1 and discussed on page 11, lines 9-30 and elsewhere in the specification] said processing task including allocating memory areas for data storage; and

suspending code operable to suspend [step 130 shown in Figure 1 and discussed on page 11, lines 9-30 and elsewhere in the specification] an actual execution path of said processing task at an execution point to perform memory management;

reference identifying code operable to identify [step 140 shown in Figure 1 and discussed on page 11, lines 9-30, step 330 shown in Figure 3 and discussed on page 16, lines 4-22, step 550 shown in Figure 5 and discussed on page 18, lines 4-23 and elsewhere in the specification] at least one data item roots occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

correlating code operable to determine [step 150 shown in Figure 1 and discussed on page 11, lines 9-30, step 340 shown in Figure 3 and discussed on page 16, lines 4-22 and elsewhere in the specification] a correlation between reference values corresponding to said identified data item root and memory areas allocated during said execution up to said execution point; and

memory management code operable to perform [step 160 shown in Figure 1 and discussed on page 11, lines 9-30 and on page 13, lines 5-17, step 350 shown in Figure 3 and discussed on page 16, lines 4-22 and elsewhere in the specification] a memory management operation on allocated memory areas in dependence upon said correlation."

"21. A data processing apparatus operable to control execution of a processing task within a data processing system, said data processing apparatus comprising:

execution logic operable to execute [step 120 shown in Figure 1 and discussed on page 11, lines 9-30 and elsewhere in the specification] said processing task including allocating memory areas for data storage; and

task suspension logic operable to suspend [step 130 shown in Figure 1 and discussed on page 11, lines 9-30 and elsewhere in the specification] an actual execution path of said processing task at an execution point to perform memory management;

reference identifying logic operable to identify [step 140 shown in Figure 1 and discussed on page 11, lines 9-30, step 330 shown in Figure 3 and discussed on page 16, lines 4-22, step 550 shown in Figure 5 and discussed on page 18, lines 4-23 and elsewhere in the specification] at least one data item occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

correlation logic operable to determine [step 150 shown in Figure 1 and discussed on page 11, lines 9-30, step 340 shown in Figure 3 and discussed on page 16, lines 4-22 and elsewhere in the specification] a correlation between reference values corresponding to identified said at least one data item root and memory areas allocated during said execution up to said execution point by identifying at least one data item reachable from said at least one data item root; and

memory management logic operable to perform [step 160 shown in Figure 1 and discussed on page 11, lines 9-30 and on page 13, lines 5-17, step 350 shown in Figure 3 and discussed on page 16, lines 4-22 and elsewhere in the specification] a memory management operation on allocated memory areas in dependence upon said correlation."

"31. A method of identifying for a memory management operation at least one data item root and at least one data item reachable from said data item root comprising the steps of:

scanning [step 610 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] a plurality of program instructions corresponding to said processing task and logging [step 620 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] a data type for each store instruction corresponding to each of said at least one data item;

categorizing [step 630 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] at least one of said at least one data item root or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

simulating [step 530 shown in Figure 5 and discussed on page 18, lines 5-23, paths 1 & 2 for VAR2 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and determining the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

checking [step 540 shown in Figure 5 and discussed on page 18, lines 10-23, path 1 and path 2 in array 630 shown in Figure 6 and discussed on page 19, lines 7-18 and elsewhere in the specification] said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management operation is performed in dependence upon a result of said step of checking [step 540 shown in Figure 5 and discussed on page 18, lines 10-23, path 1 and path 2 in array 630 shown in Figure 6 and discussed on page 19, lines 7-18 and elsewhere in the specification] said determined data type."

"32. A computer program product comprising a computer readable storage medium comprising computer readable instructions that, when executed, control a computer to identify for a memory management operation performed by memory management code at least one data item root and at least one data item reachable from said at least one data item root, wherein each of said at least one data item is a local variable, comprising:

scanning code operable to scan [step 610 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] a plurality of program instructions corresponding to said processing task and to log [step 620 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18

and elsewhere in the specification] a data type for each store instruction corresponding to each of said at least one data item;

categorizing code operable to categorize [step 530 shown in Figure 5 and discussed on page 18, lines 5-23, paths 1 & 2 for VAR2 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] at least one of said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation code operable to simulate [step 530 shown in Figure 5 and discussed on page 18, lines 5-23, paths 1 & 2 for VAR2 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking code operable to check [step 540 shown in Figure 5 and discussed on page 18, lines 10-23, path 1 and path 2 in array 630 shown in Figure 6 and discussed on page 19, lines 7-18 and elsewhere in the specification] said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management code is operable to perform said memory management operation in dependence upon a result of said data type checking code [step 540 shown in Figure 5 and discussed on page 18, lines 10-23, path 1 and path 2 in array 630 shown in Figure 6 and discussed on page 19, lines 7-18 and elsewhere in the specification]."

"33. A data processing apparatus for identifying, for a memory management operation performed by memory management logic, at least one data item root and at least one data item reachable from said at least one data item root, wherein each of said at least one data item is a local variable, comprising:

scanning logic operable to scan [step 610 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] a plurality of program instructions corresponding to said processing task and logging [step 620 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] a data type for each store instruction corresponding to each of said at least one data item;

categorizing logic operable to categorize [step 530 shown in Figure 5 and discussed on page 18, lines 5-23, paths 1 & 2 for VAR2 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] at least one of said at least one data item root or said at least one

data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation logic operable to simulate [step 530 shown in Figure 5 and discussed on page 18, lines 5-23, paths 1 & 2 for VAR2 shown in Figure 6 and discussed on page 18, line 25 to page 19, line 18 and elsewhere in the specification] all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking logic operable to check [step 540 shown in Figure 5 and discussed on page 18, lines 10-23, path 1 and path 2 in array 630 shown in Figure 6 and discussed on page 19, lines 7-18 and elsewhere in the specification] said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management logic is operable to perform said memory management operation in dependence upon a result of checking [step 540 shown in Figure 5 and discussed on page 18, lines 10-23, path 1 and path 2 in array 630 shown in Figure 6 and discussed on page 19, lines 7-18 and elsewhere in the specification] said determined data type."

## VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-6, 9, 11-16, 19, 21-26 and 29 stand rejected under 35 USC §102 as being anticipated by Wilson.

Claims 10, 20 and 30 stand rejected under 35 USC §103 as unpatentable over Wilson.

Claims 7, 8, 17, 18, 27, 28 and 31-33 stand rejected under 35 USC §103 as unpatentable over Wilson in view of Hosoya.

## VII. ARGUMENT

Appellant's arguments include the fact that the burden is on the Examiner to first and foremost properly construe the language of the claims to determine what structure and/or method steps are covered by that claim. After proper construction of the claim language, the burden is also on the Examiner to demonstrate where a single reference (in the case of anticipation) or one of a plurality of references (in the case of an obviousness rejection) teaches each of the structures and/or method steps recited in independent claims 1, 11, 21 and 31-33.

The Court of Appeals for the Federal Circuit has noted in the case of *Lindemann Maschinenfabrik GMBH v. American Hoist & Derrick*, 221 USPQ 481, 485 (Fed. Cir. 1984) that "[a]nticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim."

Furthermore, the Court of Appeals for the Federal Circuit has stated in the case of *In re Rouffet*, 47 USPQ2d 1453, 1458 (Fed. Cir. 1998)

> to prevent the use of hindsight based on the invention to defeat patentability of the invention, this court **requires** the examiner to show a **motivation** to combine the references that create the case of obviousness. In other words, the Examiner **must show reasons** that the skilled artisan, confronted with the same problems as the inventor and with no knowledge of the claimed invention, would select the elements from the cited prior art references for combination in the manner claimed. (Emphasis added).

**A. Neither Wilson nor any other prior art reference discloses "suspending an actual execution path of said processing task at an execution point"**

Each of Appellant's independent claims 1, 11 and 21 recite "suspending an **actual execution path** of said processing task at an execution point to perform memory management." (emphasis added). Independent claims 31-33 are directed to a method of identifying at least one data item root at "**said execution point**." (emphasis added). Therefore, in order to anticipate the subject matter of Appellant's independent claims 1, 11, 21 and 31-33, the burden is on the Examiner to establish where Wilson or at least one reference discloses the computer program product or apparatus for "suspending" an execution path of the processing task.

As pointed out in Appellant's previously filed Amendment (July 27, 2006 at page 19, second and third paragraphs), the Wilson reference is an example of the well known "mark-sweep" technique as discussed in Appellant's specification, page

4, lines 19-23. As pointed out in the first paragraph on page 20 of Appellant's previously filed Amendment, there is no indication in Wilson or any other cited reference of record which discloses **suspension of an actual execution path of said processing task**.

The Examiner has failed to identify any portion of Wilson (or any other reference for that matter) which discloses the above claim language, i.e., "suspending . . . ."

The Examiner would appear to concur in this analysis (and thereby admit that Wilson fails to teach this claimed element) because his only response, to the Amendment pointing out this glaring deficiency in the anticipation argument, is to argue that "it is known to one of skill in the art that garbage collection occurs at a point in an execution path where execution is suspended in order to perform garbage collection." (Final Rejection, page 9, lines 1-3). It is improper to base an anticipation rejection on a combination of Wilson and an allegation of "known to one of skill in the art."

To the extent the Examiner is now taking the position that the step of "suspending . . ." is known, that contention is respectfully traversed pursuant to §2144.03 of the Manual of Patent Examining Procedure. Section 2144.03 specifies that where "the applicant traverses such an assertion the examiner should cite a reference in support of his or her position." Appellant's previous Amendment, the first full paragraph on page 20, clearly traverses the Examiner's assertion that Wilson

teaches any suspension of execution. Yet, the Examiner has cited no other reference in support of his position that it is know to conduct garbage collection during a processing task.

It should be understood that the Wilson reference merely discloses the first stage of operation of the known "mark-sweep" garbage collectors, which involves distinguishing the live objects from the garbage by marking objects that can be traced starting a root set. Wilson (and other known methods) perform memory management operations at the bytecode-verification stage, **prior to execution of the processing task**. Appellant's claimed invention clearly does not operate prior to execution of the processing task and instead suspends the processing task at an execution point.

Because neither Wilson, nor any other cited prior art, teaches any such processing task "suspension" step and therefore Wilson cannot anticipate independent claims 1, 11 and 21 and claims dependent thereon and thus any further rejection under 35 USC §§102 or 103 is respectfully traversed.

**B. Since Wilson does not teach suspension of the processing task, he cannot teach the method step of "identifying at least one data item roots occurring in the course of execution and accessible to said processing task at said execution point"**

As noted above, Wilson operates **prior to execution of the processing task** and therefore cannot teach the third "identifying" step in Appellant's

independent claims 1, 11 and 21 which recites "**occurring in the course of execution**" (emphasis added). Independent claims 31-33 go into the details of this "identifying" step, but clearly Wilson does not and cannot teach such detail because Wilson does not teach any suspension during the processing task. Again, Wilson teaches only a mark-sweep garbage collection which steps occur prior to execution of the processing task.

The failure of Wilson or any other cited reference in teaching the "identifying" step clearly indicates that the rejections under §§102 and 103 are improper.

**C. Wilson does not teach Appellant's claimed "determining a correlation . . . by identifying at least one data item reachable from said at least one data item roots"**

Wilson does not teach "suspending" or "identifying" data item roots "in the course of execution" and therefore cannot disclose or render obvious the determination of any correlation between reference values "up to said execution point."

The Examiner has identified no portion of the Wilson reference which teaches or suggests the "determining" step in claims 1, 11 and 21 and thus cannot disclose the subsequent steps which rely upon the "determining" step. Therefore, Wilson fails to meet the requirements of 35 USC §§102 and 103 by failing to

teach all claimed method steps, computer logic or apparatus elements in one or more references.

**D. Appellant's independent claims also require the performance of "a memory management operation on allocated memory areas in dependence upon said correlation"**

Because the correlation is obtained during suspension of processing (at the claimed "execution point"), it cannot possibly be disclosed or suggested by the Wilson reference since Wilson fails to teach both the suspension step, the identifying step and the determining step. Wilson's failure to teach these claimed steps, logic or elements confirms that the memory management step, which depends upon the suspension, identifying and determining steps, renders completely unsupported the §§102 and 103 rejections.

The burden is on the Examiner to demonstrate where the claimed step, logic or element is disclosed in the cited prior art. The Examiner has failed to meet his burden to demonstrate where the "memory management" step exists in any prior art reference.

**E. The Wilson reference clearly leads one of ordinary skill in the art away from Appellant's claimed invention**

As noted above and in the previous Amendment, because Wilson clearly teaches the known "mark-sweep" type garbage collection, it would lead those of

ordinary skill in the art to perform management operations at the bytecode-verification stage, i.e., prior to execution of the processing task. The Examiner has not shown or even contended that Wilson is an example of the admitted prior art know as "mark-sweep" type garbage collection which is discussed in the present specification as known prior art (page 4, lines 19-23).

As noted in Appellant's specification (page 13, line 26 to page 14, line 2), there are numerous benefits in the present invention by beginning the processing task and then suspending the execution path and tracing the data items reachable from the roots. The advantages are that this system is less memory-intensive and less processor-intensive than the known mark-sweep garbage collection techniques such as Wilson. This is because identification of objects for garbage collection starting from the roots is performed post verification (of the bytecode) and after execution of the processing task has commenced, but only as and when required at the point at which garbage collection is initiated.

The present invention is differentiated from known garbage collection techniques that rely upon analysis of large volumes of previously stored information, the collection of which slows down execution of the program (see Appellant's specification, page 5, line 25 to page 6, line 6). The present invention enables objects for garbage collection to be identified dynamically at the point in execution of the program at which garbage collection is in fact initiated.

Thus, because Wilson clearly teaches **preprocessing** (i.e., **prior to the execution** of the processing task) garbage collection, it would lead one of ordinary skill in the art away from Appellant's claimed invention **during processing.** As a result, not only can Wilson not anticipate or render obvious Appellant's independent claims 1, 11, 21 and 31-33, but it would clearly lead one of ordinary skill in the art away from such claims. Appellant has previously made this argument and the Examiner has failed to address it or identify any evidence that rebuts this case.

**F. Wilson fails to anticipate claims 1-6, 9, 11-16, 19, 21-26 and 29 under 35 USC §102**

In view of the fact that many of the previous arguments A through E are applicable to each of the three grounds of rejection, rather than being repeated, they will be incorporated by reference to the section heading letter.

As noted in the above quoted *Lindemann* case, every claimed structure must be shown in the single Wilson reference. Sections A-D identify four separate and distinct method steps, logic elements and structural elements which are missing from independent claims 1, 11 and 21. The Examiner has failed to meet his burden of showing where **any** of these four steps/logic/elements are described let alone shown where they all are described in Wilson.

As a result of the failure to demonstrate where the claimed elements are in the independent claims 1, 11, and 21, there is no basis for rejection under 35 USC §102 of claims 1, 11 and 21 or claims 2-6 & 9, 12-16 & 19, and 22-26 & 29, respectively, dependent thereon.

**G. Wilson fails to render obvious claims 10, 20 and 30 under 35 USC §103**

As noted in the previous section (Section F), four recited elements in each of the independent claims is missing from the Wilson reference. The Board is reminded that the Court of Appeals for the Federal Circuit has held that "the PTO has the burden under Section 103 to establish a *prima facie* case of obviousness." *In re Fine*, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988). The PTO "can satisfy this burden only by showing some objective teaching in the prior art or that knowledge generally available to one of ordinary skill in the art would lead that individual to combine the relevant teachings of the references." The Examiner's failure to demonstrate where Wilson teaches any one of the four identified steps/logic/elements from independent claims 1, 11 and 21, obviates the rejection of claims 10, 20 & 30 dependent thereon, respectively.

Moreover, as demonstrated in Section E above, the Wilson reference would lead those of ordinary skill in the art away from the claimed invention. The Federal Circuit has also held that it is "error to find obviousness where references

'diverge from and teach away from the invention at hand'." *Id.* As noted above, the Wilson reference teaching of the "mark-space" method of garbage collection prior to processing teaches a solution to the problem that is dramatically different from the claimed invention.

The Examiner's failure to establish a *prima facie* case of obviousness for claims 10, 20 & 30 results in failure of the rejection under 35 USC §103.

**H. Wilson combined with Hosoya fails to render obvious claims 7, 8, 17, 18, 27, 28 and 31-33 under 35 USC §103**

The comments regarding the Wilson patent in Sections F and G are herein incorporated by reference. The Examiner makes no allegation that the four steps/logic/elements which are alleged by Appellant to be missing from Wilson, are somewhere disclosed in Hosoya. Thus, even if Wilson were combined with Hosoya, there still remains no disclosure of four claimed steps/logic/elements in each of the independent claims which is missing from the combination.

Even if the four missing steps/logic/elements were somewhere disclosed, the Examiner has identified no "reason" or "motivation" for combining the references in the manner of the claimed invention, either in the independent claims including claims 31-33 or in claims 7, 8, 17, 18, 27, 28 dependent on claims 1, 11 and 21, respectively.

Finally, even if there were some suggestion for combining bits and pieces of the Wilson and Hosoya references in the manner of Appellants claims, as noted in Section E above, Wilson would lead one of ordinary skill in the art away from the claimed invention. The Federal Circuit has also opined that it is "error to find obviousness where references 'diverge from and teach away from the invention at hand'." *In re Fine* at 1099.

The fact that Wilson teaches away from the combination claimed in independent claims 1, 11 and 21, demonstrates the unobviousness of these claims and claims dependent thereon.

## VIII. CONCLUSION

As noted in detail above, the Wilson reference fails to teach Appellant's independent claims 1, 11 and 21 (and claims dependent thereon) steps of **"suspending," "identifying," "determining"** and **"performing a memory management operation."** As noted above, Wilson fails to disclose the details of the "identifying" step **during a processing task at an execution point** as set out in independent claims 31-33. Moreover, it has clearly been demonstrated that Wilson teaches garbage management operations **prior to execution** of the processing task which is the **direct opposite** of Appellant's management operation **during a processing task**. Notwithstanding the Examiner's completely unsupported allegations that the claimed invention is known to one of ordinary

skill in the art, he has cited no reference and provided no evidence in support of his conclusion.

As a result of the above, there is simply no support for the rejection of Appellant's independent claims 1, 11, 21 and 31-33 or claims dependent thereon under 35 USC §102 or §103. Thus, and in view of the above, the rejection of claims 1-33 under 35 USC §§102 and 103 is clearly in error and reversal thereof by this Honorable Board is respectfully requested.

Respectfully submitted,

**NIXON & VANDERHYE P.C.**

By: _____

Stanley C. Spooner
Reg. No. 27,393

SCS:kmm
Enclosure

## IX. CLAIMS APPENDIX

1. A method of controlling execution of a processing task within a data processing system, said method comprising the steps of:

executing said processing task including allocating memory areas for data storage; and

suspending an actual execution path of said processing task at an execution point to perform memory management, said memory management comprising the steps of:

identifying at least one data item roots occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

determining a correlation between reference values corresponding to said at least one data item roots and memory areas allocated during said execution up to said execution point by identifying at least one data item reachable from said at least one data item roots; and

performing a memory management operation on allocated memory areas in dependence upon said correlation.

2. A method as claimed in claim 1, wherein each of said at least one data items is an operand.

3. A method as claimed in claim 2, wherein said identifying step comprises:

identifying a possible execution path leading to said execution point, wherein said possible execution path may be different from said actual execution path;

performing a simulated execution of said possible execution path; and

wherein said at least one data item roots and said at least one data item accessible to said processing task are identified by following said possible execution path to said current execution point.

4. A method as claimed in claim 3, wherein said memory management operation comprises marking all of said memory areas that are accessible to said processing task either directly or indirectly through said identified data items and collecting unmarked memory areas for re-allocation during subsequent execution of said processing task.

5. A method as claimed in claim 4, wherein said memory management operation comprises compacting said unmarked memory areas prior to re-allocation.

6. A method as claimed in claim 1, wherein each of said at least one data item is a local variable.

7. A method as claimed in claim 6, wherein said identifying step comprises:

scanning a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said at least one data items;

categorising at least one of said data item roots or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

simulating all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorised as a multiple-type variable and determining the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

checking said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point; and

said memory management operation is performed in dependence upon a result of said step of checking said determined data type.

8. A method as claimed in claim 7, wherein said memory management operation involves tagging said at least one data item as suitable for reallocation if

said determined data type is different for different ones of said possible execution paths at said current execution point.

9. A method as claimed in claim 1, wherein said processing task is a component of a computer program written in an object-oriented programming language.

10. A method as claimed in claim 9, wherein said object-oriented programming language is Java.

11. A computer program product comprising a computer readable storage medium comprising computer readable instructions that, when executed, cause a computer to control execution of a processing task within a data processing system, said computer program comprising:

execution code operable to execute said processing task including allocating memory areas for data storage; and

suspending code operable to suspend an actual execution path of said processing task at an execution point to perform memory management;

reference identifying code operable to identify at least one data item roots occurring in the course of execution and accessible to said processing task at said

execution point which specify reference values pointing to respective ones of said memory areas;

correlating code operable to determine a correlation between reference values corresponding to said identified data item root and memory areas allocated during said execution up to said execution point; and

memory management code operable to perform a memory management operation on allocated memory areas in dependence upon said correlation.

12. A computer program product as claimed in claim 11, wherein each of said at least one data item is an operand.

13. A computer program product as claimed in claim 12, wherein said reference identifying code comprises:

path identifying code operable to identifying a possible execution path leading to said execution point, wherein said possible execution path may be different from said actual execution path;

path simulation code operable to perform a simulated execution of said possible execution path; and

wherein said at least one data item root and said at least one data item accessible to said processing task are identified by following said possible execution path to said current execution point.

14. A computer program product as claimed in claim 13, wherein said memory management code is operable to mark all of said memory areas that are accessible to said processing task either directly or indirectly through said identified data items and to collect umarked memory areas for re-allocation during subsequent execution of said processing task.

15. A computer program product as claimed in claim 14, wherein said memory management code is operable to compact said unmarked memory areas prior to re-allocation.

16. A computer program product as claimed in claim 11, wherein each of said one or more data items is a local variable.

17. A computer program product as claimed in claim 16, wherein said reference identifying code comprises:

scanning code operable to scan a plurality of program instructions corresponding to said processing task and to log a data type for each store instruction corresponding to each of said one or more data items;

categorising code operable to categorise at least one of said one or more data items as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation code operable to simulate all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorised as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking code operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point; and

wherein said memory management code is operable to perform said memory management operation in dependence upon a result of said data type checking code.

18. A computer program product as claimed in claim 17, wherein said memory management code is operable to tag said at least one data item as suitable for reallocation if said determined data type is different for different ones of said possible execution paths at said current execution point.

19. A computer program product as claimed in claim 11, wherein said processing task is a component of a computer program written in an object-oriented programming language.

20. A computer program product as claimed in claim 19, wherein said object-oriented programming language is Java.

21. A data processing apparatus operable to control execution of a processing task within a data processing system, said data processing apparatus comprising:

execution logic operable to execute said processing task including allocating memory areas for data storage; and

task suspension logic operable to suspend an actual execution path of said processing task at an execution point to perform memory management;

reference identifying logic operable to identify at least one data item occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

correlation logic operable to determine a correlation between reference values corresponding to identified said at least one data item root and memory

areas allocated during said execution up to said execution point by identifying at least one data item reachable from said at least one data item root; and

memory management logic operable to perform a memory management operation on allocated memory areas in dependence upon said correlation.

22. A data processing apparatus as claimed in claim 21, wherein each of said one or more data items is an operand.

23. A data processing apparatus as claimed in claim 22, wherein said reference identifying logic comprises:

path identifying logic operable to identify a possible execution path leading to said execution point, wherein said possible execution path may be different from said actual execution path;

path simulation logic operable to perform a simulated execution of said possible execution path; and

wherein said at least one data item and said at least one data item root accessible to said processing task are identified by following said possible execution path to said current execution point.

24. A data processing apparatus as claimed in claim 23, wherein said memory management logic is operable to mark all of said memory areas that are

accessible to said processing task either directly or indirectly through said identified data items and collecting umarked memory areas for re-allocation during subsequent execution of said processing task.

25. A data processing apparatus as claimed in claim 24, wherein said memory management logic is operable to compact said unmarked memory areas prior to re-allocation.

26. A data processing apparatus as claimed in claim 21, wherein each of said at least one data item is a local variable.

27. A data processing apparatus as claimed in claim 26, wherein said reference identifying logic comprises:

scanning logic operable to scan a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said one or more data items;

categorising logic operable to categorise at least one of said at least one data item roots or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation logic operable to simulate all possible execution paths up to said execution point for each of said at least one data itemroot or said at least one

data item categorised as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking logic operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point; and

said wherein memory management logic is operable to perform said memory management operation in dependence upon a result of checking said determined data type.

28. A data processing apparatus as claimed in claim 27, wherein said memory management logic is operable to tag said data item as suitable for reallocation if said determined data type is different for different ones of said possible execution paths at said current execution point.

29. A data processing apparatus as claimed in claim 21, wherein said processing task is a component of a computer program written in an object-oriented programming language.

30. A data processing apparatus as claimed in claim 29, wherein said object-oriented programming language is Java.

31. A method of identifying for a memory management operation at least one data item root and at least one data item reachable from said data item root comprising the steps of:

scanning a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said at least one data item;

categorizing at least one of said at least one data item root or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

simulating all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and determining the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

checking said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management operation is performed in dependence upon a result of said step of checking said determined data type.

32. A computer program product comprising a computer readable storage medium comprising computer readable instructions that, when executed, control a computer to identify for a memory management operation performed by memory management code at least one data item root and at least one data item reachable from said at least one data item root, wherein each of said at least one data item is a local variable, comprising:

scanning code operable to scan a plurality of program instructions corresponding to said processing task and to log a data type for each store instruction corresponding to each of said at least one data item;

categorizing code operable to categorize at least one of said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation code operable to simulate all possible execution paths up to -said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking code operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management code is operable to perform said memory management operation in dependence upon a result of said data type checking code.

33. A data processing apparatus for identifying, for a memory management operation performed by memory management logic, at least one data item root and at least one data item reachable from said at least one data item root, wherein each of said at least one data item is a local variable, comprising:

scanning logic operable to scan a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said at least one data item;

categorizing logic operable to categorize at least one of said at least one data item root or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation logic operable to simulate all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking logic operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management logic is operable to perform said memory management operation in dependence upon a result of checking said determined data type.

## X. EVIDENCE APPENDIX

None.

## XI. RELATED PROCEEDINGS APPENDIX

None.